IS 6733: Deep Learning
Homework 4

P1 (35pt): Write a Python code using NumPy, Matplotlib, and Keras to perform image classification for the Fashion_MINIST datset (https://github.com/zalandoresearch/fashion-mnist)

1. (5pt) Load the dataset using tf.keras.datasets.fashion_mnist.load_data() and show the first 12 images of the training dataset in two rows.
2. (5pt) Add the "depth" dimension to the training/testing image data using .reshape(), use to_categorical() to transform all labels into their one-hot encoding forms, and normalize the pixel values of all images into [0, 1]. Print out the shapes of training and testing images.
   - Note that the imported training/testing image data have a shape of (number_samples, image_height, image_width) and you want to reshape it into the shape of (number_samples, image_height, image_width, image_depth/image_channels)
3. (10pt) Build a CNN model using a stack of Conv2D (128 filters of size (3, 3) with ReLU activation), MaxPooling2D (pool size of (2, 2)), Conv2D (64 filters of size (3, 3 with ReLU activation), MaxPooling2D (pool size of (2, 2)), Dense (128 hidden units with ReLU activation), and output layer. Display the model architecture using .summary().
   - You need to specify other parameters of the input layer and output layer.
4. (10pt) Compile and train the model for 10 epochs and batch size of 32. Set verbose = 0 during the training to compress the training progress. Draw the plot of the training accuracy w.r.t. the epoch number
   - You need to specify the right optimizer, loss function, and metrics for this task.
5. (5pt) Test your trained model on the testing dataset and observe the loss and accuracy using .evaluate().

P2 (65pt): Write a Python code using NumPy, Matploblib and Keras to perform image classification using pre-trained model for the CIFAR-10 dataset (https://www.cs.toronto.edu/~kriz/cifar.html).

1. (5pt) Load the dataset using tf.keras.datasets.cifar10.load_data() and show the first 20 images of the training dataset in two rows.
   - You will obtain the pair of feature matrix and label vector for the training dataset and the pair of feature matrix and label vector for the testing dataset at the end of this step
   - Note that the CIFAR-10 dataset consists of 60000 32x32 color images in 10 classes, each with a label value within [0, 9]. In the following step, we want to partition this dataset into two training/testing pairs, one containing images with labels in [0, 4] and the other containing images with labels in [5, 9].
2. (5 pt) Reshape the label vectors in both the training and testing datasets to 1D using .reshape(), and compare them with 5 to find out the indices of images that have class labels < 5 and class labels >= 5, respectively, in the training and testing datasets.
   - You will obtain four index arrays of Boolean values at the end of this step (<5 and >= 5 for training dataset and <5 and >=5 for testing dataset)
   - Hint: label_vector < 5 and label_vector >= 5 will generate such indices

3. (5 pt) Use the index arrays obtained in the previous step to split the training/testing dataset into two subsets (each consisting of a feature matrix and a label vector): one with class labels < 5 and one with class labels >= 5. Print out the shapes of the resulting subsets for both training and testing datasets.
    - You will obtain four subsets at the end of this step: one pair of training and testing subsets of images with class labels < 5 and another pair of training and testing subsets of images with class labels >= 5.
4. (5pt) Subtract 5 from the label vectors of the pair of training and testing subsets with class labels >= 5 so that the label vectors in this pair of subsets contains values from 0 to 4. Use `to_categorical()` to transform all labels into their one-hot encoding forms, and normalize the pixel values of all images into [0, 1].
5. (5pt) Build a CNN `model_1` using a stack of Conv2D (64 filters of size (3, 3) with ReLU activation), Conv2D (64 filters of size (3, 3) with ReLU activation), MaxPooling2D (pool size of (2, 2)), Dense (128 hidden units with ReLU activation), and output layer. Display the model architecture using `.summary()`.
    - You need to specify the correct hyperparameters of the input layer and output layer.
6. (10pt) Compile and train the model on the subset of training images with class labels < 5 for 20 epochs and batch size of 128. Set verbose = 0 during the training to compress the results. Draw the plot of the training accuracy w.r.t. the epoch number.
    - You need to specify the correct optimizer, loss function, and metrics for this task.
7. (5pt) Test your trained `model_1` on the subset of testing images with class labels <5 and observe the loss and accuracy using `.evaluate()`.
8. (10pt) Build a new CNN `model_2` that has the same architecture as `model_1` and reuse the pre-trained convolutional base layers of `model_1` (i.e., all layers before applying flatten()). You need to freeze the pre-trained convolutional base layers of `model_2` so that their model parameters will not be changed during the training. Display the model architecture of `model_2` using `.summary()`.
    - One method to achieve the above step is as follows (You can use other methods as long as they achieve the same goal):

```
#Clone a new model from an existing model
model_2 = keras.models.clone_model(model_1)

# Freeze all layers except the last two layers.
for layer in model_2.layers[:-2]:
  layer.trainable = False

model_2.summary()
```

9. (10pt) Compile `model_2`, and train it on the subset of training images with class labels >=5 for 20 epochs and batch size of 128. Draw the plot of the training accuracy w.r.t. the epoch number.
10. (5pt) Test your trained `model_2` on the subset of testing images with class labels >=5 and observe the loss and accuracy using `.evaluate()`.

Submission Instruction: Submit a PDF file of your codes and outputs and a Google Colab shared link to your source file (.ipynb format) to Blackboard.